KUL-Eval: A Combinatory Categorial Grammar Approach for Improving Semantic Parsing of Robot Commands using Spatial Context

Willem Mattelaer, Mathias Verbeke and Davide Nitti
Department of Computer Science, KU Leuven, Belgium

Abstract

When executing commands, a robot has a certain level of contextual knowledge about the environment in which it operates. Taking this knowledge into account can be beneficial to disambiguate commands with multiple interpretations. We present an approach that uses combinatory categorial grammars for improving the semantic parsing of robot commands that takes into account the spatial context of the robot. The results indicate a clear improvement over non-contextual semantic parsing. This work was done in the context of the SemEval-2014 task on supervised semantic parsing of spatial robot commands.

1 Introduction

One of the long-standing goals of robotics is to build autonomous robots that are able to perform everyday tasks. Two important requirements to achieve this are an efficient way of communicating with the robot, and transforming these commands such that the robot is able to capture their meaning. Furthermore, this needs to be consistent with the context in which the robot is operating, i.e., the robot's *belief*.

Semantic parsing focuses on translating natural language (NL) into a formal representation that captures the meaning of the sentence. Most of the current semantic parsing approaches are non-contextual, i.e., they do not take into account the context in which the command sentence should be executed. This can lead to erroneous parses, most often due to ambiguity in the original sentence. Consider the following example sentence *"Move the pyramid on the blue cube on the gray cube"*. This sentence has two valid interpretations. Either the robot needs to move the pyramid that is currently standing on the blue cube and put it on the gray cube, or move the pyramid and place it on the blue cube that is standing on the gray cube.

Humans will decide on the correct interpretation by taking into account the context. For instance, by looking at Figure 1, it is clear that the second interpretation is not possible, because there is no blue cube on top of a gray cube. However, there is a pyramid on top of a blue cube, making the first interpretation possible. The goal of this paper is to improve on non-contextual semantic parsing by tailoring the context to guide the parser. In this way, part of the ambiguity that causes multiple interpretations can be resolved.

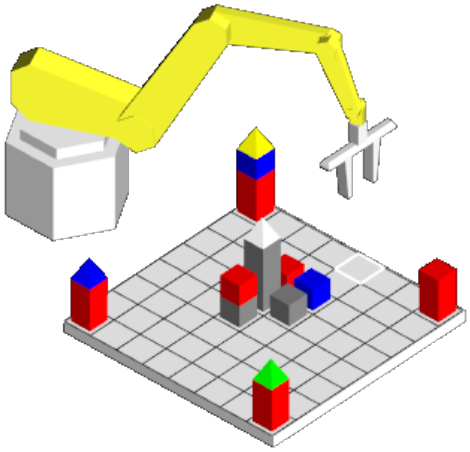

Figure 1: Possible situation (taken from (Dukes, 2013b))

Our approach consists of two steps. First, non-contextual semantic parsing using combinatory categorial grammars (CCG) (Steedman, 1996; Steedman, 2000) is performed on the sentence. This returns multiple possible parses, each with an attached likelihood of correctness. Subsequently, each parse is checked against the current context. The parse with the highest score that is possible given the current context is returned.

This paper is organized as follows. In Section 2 we discuss related work, followed by a detailed description of our approach in Section 3. In Section 4, the approach is evaluated and compared to

non-contextual parsing. Finally, in Section 5 we conclude and outline directions for future work.

The software is available from `https://github.com/wmattelaer/Thesis`.

2 Related Work

There is a significant body of previous work on learning semantic parsers. We will first review approaches that translate NL sentences into a formal representation without taking context into account, followed by related techniques that use the context to improve the parsing.

Our approach is inspired by the work of Kwiatkowski et al. (2010). The authors present a supervised CCG approach to parse queries to a geographical information database and a flight-booking system. This differs from the current setting in that the database querying does not require to take the context of the environment into account, as is the case when executing robot commands. SILT (Kate et al., 2005) uses transformation rules to translate the NL sentence to a query for the robot. This approach was extended to tailor support vector machines with string kernels (KRISP) (Kate and Mooney, 2006) and statistical machine learning (WASP) (Mooney, 2007). Also unsupervised approaches exist. Poon (2013) solves this lack of supervision by 1) inferring supervision using the target database, which constrains the search space, and 2) by using augmented dependency trees.

Artzi and Zettlemoyer (2013) study the use of grounded CCG semantic parsing using weak supervision for interpreting navigational robot commands. Their approach is similar to ours, but instead of postprocessing the results in a verification step, the context (or state) is added to the training data. Krishnamurthy and Kollar (2013) use CCGs as a foundation, but match it to the context using an evaluation function. This evaluation function scores a denotation, i.e., the set of entity referents for the entire sentence, given a logical form and a knowledge base, which is considered as the context.

3 Methodology

Our approach consists of two steps: a parse step and a verification step. Before these steps can be executed, a Combinatory Categorial Grammar needs to be trained. The training data for this grammar consists of typed λ-expressions (Carpenter, 1997) that are annotated with their corresponding NL sentences. As the input data for the SemEval-2014 task consists of Robot Control Language (RCL) expressions (Dukes, 2013a)[1], the data needs to be preprocessed first.

3.1 Preprocessing

During preprocessing, the RCL expressions are transformed into equivalent λ-expressions. In the λ-expressions, each entity is represented by a lambda term where the variable is a reference to the object. The properties of an entity are defined by a conjunction of literals with two arguments. The predicate details the property that is being defined. An example entity, a blue cube, can be represented as $\lambda x.color(x, blue), type(x, cube)$. A spatial relation between two entities is a literal with three arguments: the variable of the first entity, the type of relation and the second entity. The latter is given by its lambda term. This lambda term has to be wrapped in a definite determiner, det, that selects a single element from the set created by the lambda term (Artzi and Zettlemoyer, 2013). For example: the RCL expression

```
(entity:
    (type: prism)
    (spatial-relation:
        (relation: above)
        (entity:
            (color: blue)
            (type: cube))))
```

is transformed to the λ-calculus expression

$$\lambda x.type(x,prism), relation(x, above, det(\lambda y.color(y, blue), type(y, cube)))$$

Events are contained in one lambda term with one variable per event. There are three possible event predicates. The $action$ predicate defines the action by detailing the action type and the object entity. The $destination$ predicate will set the destination of the object[2]. Finally, the $sequence$ predicate is necessary to detail the order of the events. An example of this can be seen at the bottom of Figure 2.

Besides transforming the RCL expressions to λ-calculus, also the action types of the events are checked. If an event has an action of type $move$ or $drop$, it is changed to the combined $move$ &

[1] RCL is a linguistically-oriented formal language for controlling a robot arm, that represents entities, attributes, anaphora, ellipsis and qualitative spatial relations.

[2] Note that this event is not always necessary, e.g., in the case of a $take$ action, the robot will not release the object.

$drop$ action type. This change was introduced because the actual verbs that are used to instruct the robot to perform one of these two actions are often the same. To illustrate this, consider the following two sentences taken from the training data: *"place blue block on top of single red block"* and *"place green block on top of blue block"*. In the former, the intended action is a $drop$ action, while in the latter the action should be a $move$ action. During parsing, the correct action can be selected by looking at the context it has to be executed in. If the robot is currently grasping an object, the intended action is a $drop$ action, otherwise it is a $move$ action.

Furthermore, the anaphoric references are resolved in the natural language sentences. Anaphoric references are words that refer to one or more words mentioned earlier in the sentence. The sentences of the dataset are annotated with markers that capture the references in the sentence. The markers that are used are `[1]`, `(1)` and $\{1\}$ and are placed right after the word that is used for the reference. `[1]` is used to mark a word that is referred to by another word, whereas `(1)` is used to detail a word that refers to another word, e.g., *it*. Finally, $\{1\}$ marks a word that refers to the type of an earlier entity, e.g., *one*. The numbers in these markers can increase if there are different references in one sentence, but the sentences of this dataset do not contain different references. For instance, the sentence *Pick the blue block and place it above the gray one* is transformed to the sentence *Pick the blue block [1] and place it (1) above the gray one* $\{1\}$.

The anaphoric references are found using the coreference resolution system of *Stanford CoreNLP* (Recasens et al., 2013; Lee et al., 2013; Lee et al., 2011; Raghunathan et al., 2010). However, it is not capable of finding references that use *one*. This can be solved by letting the *one* always refer to the first entity of the sentence, because of the simplicity of the sentences.

3.2 Parsing

To parse the robot commands, a Probabilistic Combinatory Categorial Grammar (PCCG) (Kwiatkowski et al., 2010) is used. Regular CCGs consist out of two sets: a lexicon of lexical items and a set of operations. A lexical entry combines a word or phrase with its meaning. This meaning is represented by a category. A category captures the syntactic as well as the the semantic information of a word. A number of primitive symbols, a subset of the part-of-speech tags, are used to represent the syntax. These primitive symbols can be combined using specific operator symbols ($/$, $\backslash$). The semantics are represented by a λ-expression. Some example lexical entries are:

blue $\vdash ADJ : \lambda x.color(x, blue)$
pyramid $\vdash N : \lambda x.type(x, prism)$
pick up $\vdash S/NP : \lambda y \lambda x.action(x, take, y)$

The operator symbols can now be used to determine how the categories can be combined using operations. The operations that are used by the CCG take one or two categories as input and return one category as output. These operations will simultaneously address syntax and semantics. The two most frequently used operations are the application operations, i.e., *forward* $(>)$ and *backward* $(<)$:

$$X/Y : f \quad Y : g \Rightarrow X : f(g) \quad (>)$$
$$Y : g \quad X\backslash Y : f \Rightarrow X : f(g) \quad (<)$$

The forward application takes as input a CCG category with syntax X/Y and λ-expression f followed by a category with syntax Y and λ-expression g and returns a CGG category with syntax X and λ-expression $f(g)$.

The operations will derive syntactic and semantic information, while keeping track of the word order that is encoded using the slash direction.

Another important operation deals with the definite determiner in the λ-expressions:

$$N : f \Rightarrow NP : det(f)$$

This operation takes a single noun (N) category as input and returns an noun phrase (NP) category where the original λ-expression is wrapped in a determiner. A complete parsing example is shown in Figure 2.

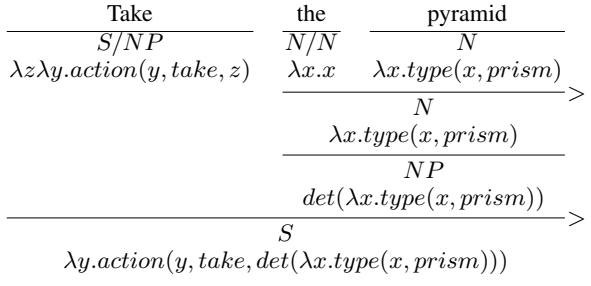

Figure 2: A possible parse for the sentence "Take the pyramid"

CCGs will usually have multiple possible parses for a sentence given a certain lexicon for which it is not possible to determine which of these is best. To alleviate this problem, PCCGs have been introduced (Kwiatkowski et al., 2010). PCCGs will return the most likely parse using a log-linear model that contains a parameter vector θ, estimated using stochastic gradient updates. The joint probability of a λ-calculus expression z and a parse y is given by $P(y, z|x; \theta, \Lambda)$, with Λ being the entire lexicon. The most likely λ-calculus expression z given a sentence x can then be found by:

$$f(x) = \arg\max_z P(z|x; \theta, \Lambda)$$

where the probability of z is equal to the sum of the probabilities of all parses that produce z:

$$P(z|x; \theta, \Lambda) = \sum_y P(y, z|x; \theta, \Lambda)$$

For training the PCCGs, the algorithm as described by Kwiatkowski et al. (2010) was used. It consists of two steps. In the first step the lexicon is expanded with new lexical items. The second step will update the parameters of the grammar using stochastic gradient updates (LeCun et al., 1998). All parameters are associated with a feature. The system uses *lexical features*: for each item in the lexicon a feature is added that fires when the item is used.

3.3 Verification

The parser will return multiple λ-expressions, each with an attached likelihood score. In the verification step, these resulting expressions are checked against the context. These λ-expressions are first transformed to RCL expressions[3]. Next, the entities are extracted from the RCL expressions and for each entity a corresponding object is searched using a spatial planner, provided by the task organizer. This spatial planner will, given an entity description in RCL, return the objects in the context that satisfy that description. RCL expressions with entities that have no corresponding object in the context are discarded. From the remaining RCL expressions the one with the highest likelihood is returned.

[3]Note that during pre- and postprocessing no information is lost, as the mapping between λ-calculus and RCL is a one-to-one function.

	Complete	Partial	Without context
Correct	71.29%	78.58%	57.76%
Wrong	11.66%	4.37%	27.72%
No result	17.05%	17.05%	14.52%

Table 1: Results

4 Evaluation

The provided dataset for the task was crowdsourced using Train Robots, an online game in which players were given before and after images of a scene and were asked to give the NL command that the robot had executed (Dukes, 2013a). Each scene is a formal description of a discrete 8x8x8 3D game board consisting of colored blocks. The entire dataset consists of 3409 annotated examples, and was split in a training and test set of 2500 and 909 sentences respectively.

The results are listed in Table 1. The first column ("Complete") contains the results when the resulting RCL expression is exactly the same as the ground-truth RCL expression. Next to the full matching scores, we also provide the scores for partial matching of the RCL expressions ("Partial"), based on the Parseval metric (Black et al., 1991). Each RCL expression is scored between 0 and 1 according to the resemblance with the expected expression. The tree representations of the RCL expressions are compared and the number of correct nodes in the actual expression are divided by the number of nodes in the tree of the expected expression to calculate the score. A node is correct if it is present at the same position in both trees and if all children are correct.

The last column ("Without context") contains the results when using the parser without the verification step. This can be considered a baseline.

It may be clear that the use of contextual parsing is advantageous when comparing the contextual with the non-contextual setting, with an increase of 13% in the number of correct results.

Error Analysis

When inspecting the wrong parses, it could be observed that the wrong results were usually minimally wrong. Either the value of a certain element was wrong, an unnecessary element was added to the expression or a required element was not present in the resulting expression. This is also clear when comparing the complete with the partial match results, from which it can be seen that 66 sentences were only partially incorrect. Some

Expected	Actual	Occurrences
edge	region	17
above	within	15
right	left	8
left	front	7
within	above	6

Table 2: Wrong values

of the most commonly wrong values are listed in Table 2. A final common reason for a wrong parse was that a sequence of a $take$ and a $drop$ action is considered as a single $move$ action. There are 6 occurrences of this final case of which 5 would result in the same end state.

One of the most common reasons that the parser returned no result for a sentence, is because one type of sentences was not present in the training set. Sentences of the form *"pick up red block. put it on grey block"* were completely absent from the training data, but did appear 34 times in the test set. Their structure is quite simple and should not present a problem, but the parser was only trained on sentences that combined the two actions with an "and" connective. This is a problem because the trained grammar is very dependent on the provided training data. Another difficult type of sentences are the ones that contain measures. Only 17 of these were parsed correctly, while 70 had no result and 3 were wrong.

Without considering the context, the combined $move$ & $drop$ action is not possible, since the context is required to decide afterwards which specific action has to be executed. 59 sentences (6.5%) were wrong because a wrong action was selected.

5 Conclusions and Future Work

In this paper we have presented an improved semantic parsing approach for robot commands by integrating spatial context. It consists of two steps. First, the sentence is parsed using a Probabilistic Combinatory Categorial Grammar. Next, the parses are checked against the context. The resulting parse is the one with the highest likelihood that is valid given the context. This approach was evaluated on the SemEval-2014 Task 6 dataset. The results indicate that integrating contextual knowledge is advantageous for parsing spatial robot commands.

In future work, we will perform an in-depth analysis of our system in comparison with the other participating systems. Furthermore, we will extend our approach to contexts that also contain probabilistic facts, in order to be able to handle noisy sensor data.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62.

Ezra Black, Steven P. Abney, D. Flickenger, Claudia Gdaniec, Ralph Grishman, P. Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith L. Klavans, Mark Liberman, Mitchell P. Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *HLT*. Morgan Kaufmann.

Bob Carpenter. 1997. *Type-Logical Semantics*. The MIT Press.

Kais Dukes. 2013a. Semantic Annotation of Robotic Spatial Commands. In *Language and Technology Conference*.

Kais Dukes. 2013b. Supervised semantic parsing of robotic spatial commands. `http://alt.qcri.org/semeval2014/task6/`.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 913–920, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, AAAI'05, pages 1062–1068. AAAI Press.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly Learning to Parse and Perceive : Connecting Natural Language to the Physical World. In *Transactions of ACL*, volume 1, pages 193–206.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1512–1523, Stroudsburg, PA, USA. Association for Computational Linguistics.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the CoNLL-11 Shared Task*.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4).

Raymond J. Mooney. 2007. Learning for semantic parsing. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 4394 of *Lecture Notes in Computer Science*, pages 311–324. Springer Berlin Heidelberg.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *ACL (1)*, pages 933–943. Association for Computer Linguistics.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. EMNLP-2010, Boston, USA.

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013. The life and death of discourse entities: Identifying singleton mentions. In *Proceedings of NAACL 2013*, pages 627–633. Association for Computational Linguistics.

Mark Steedman. 1996. *Surface structure and interpretation*. Linguistic inquiry monographs. The MIT Press, Cambridge, MA, USA.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA, USA.